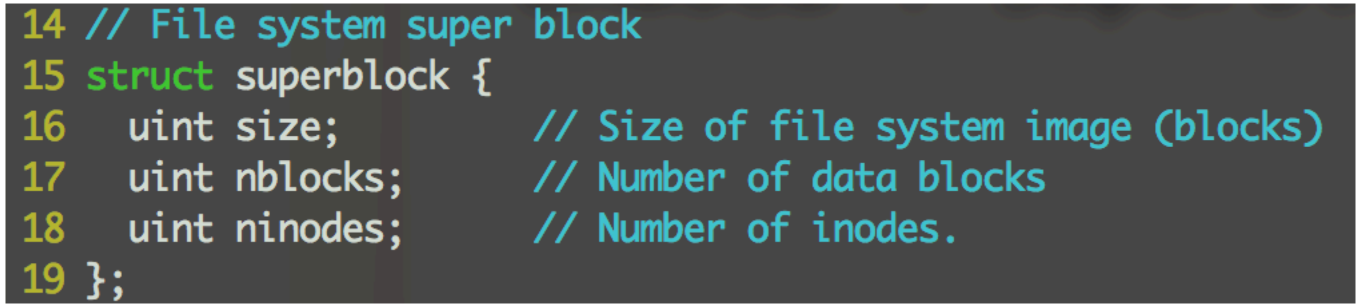
# Xv6 File System

Zhewen Song

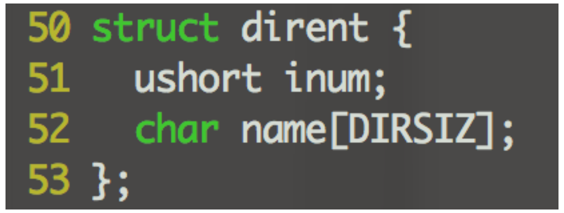
11/28/2017

# File System Layout in xv6

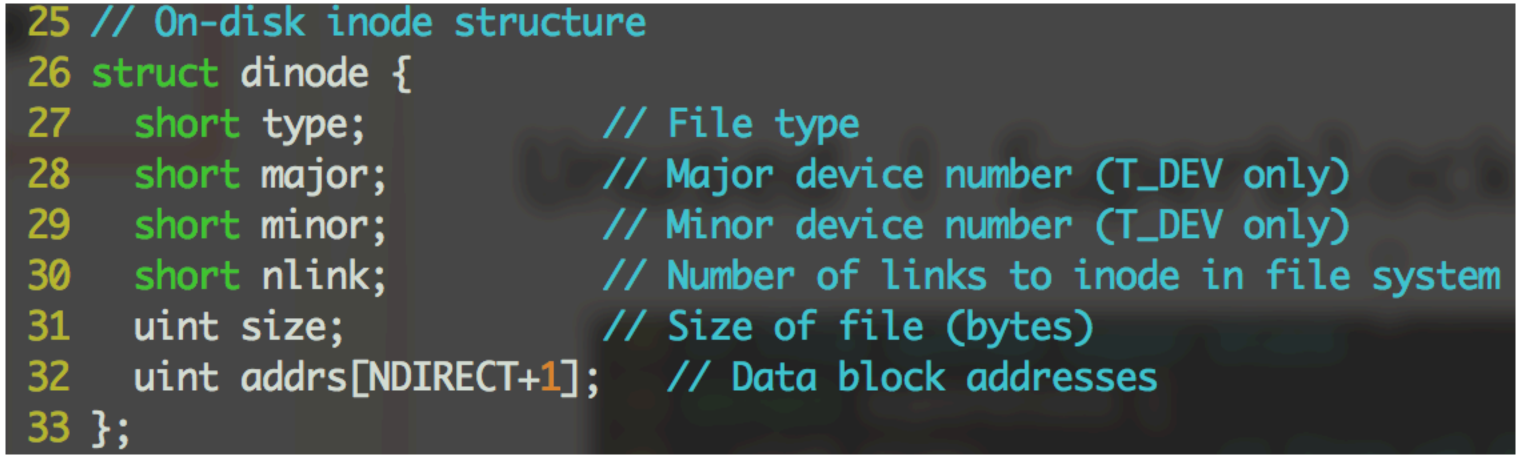
```
Unused | Superblock | Inodes ... | Unused | Bitmap | Data ...
```



```
// File system super block
struct superblock {
  uint size;         // Size of file system image (blocks)
  uint nblocks;      // Number of data blocks
  uint ninodes;      // Number of inodes.
};
```

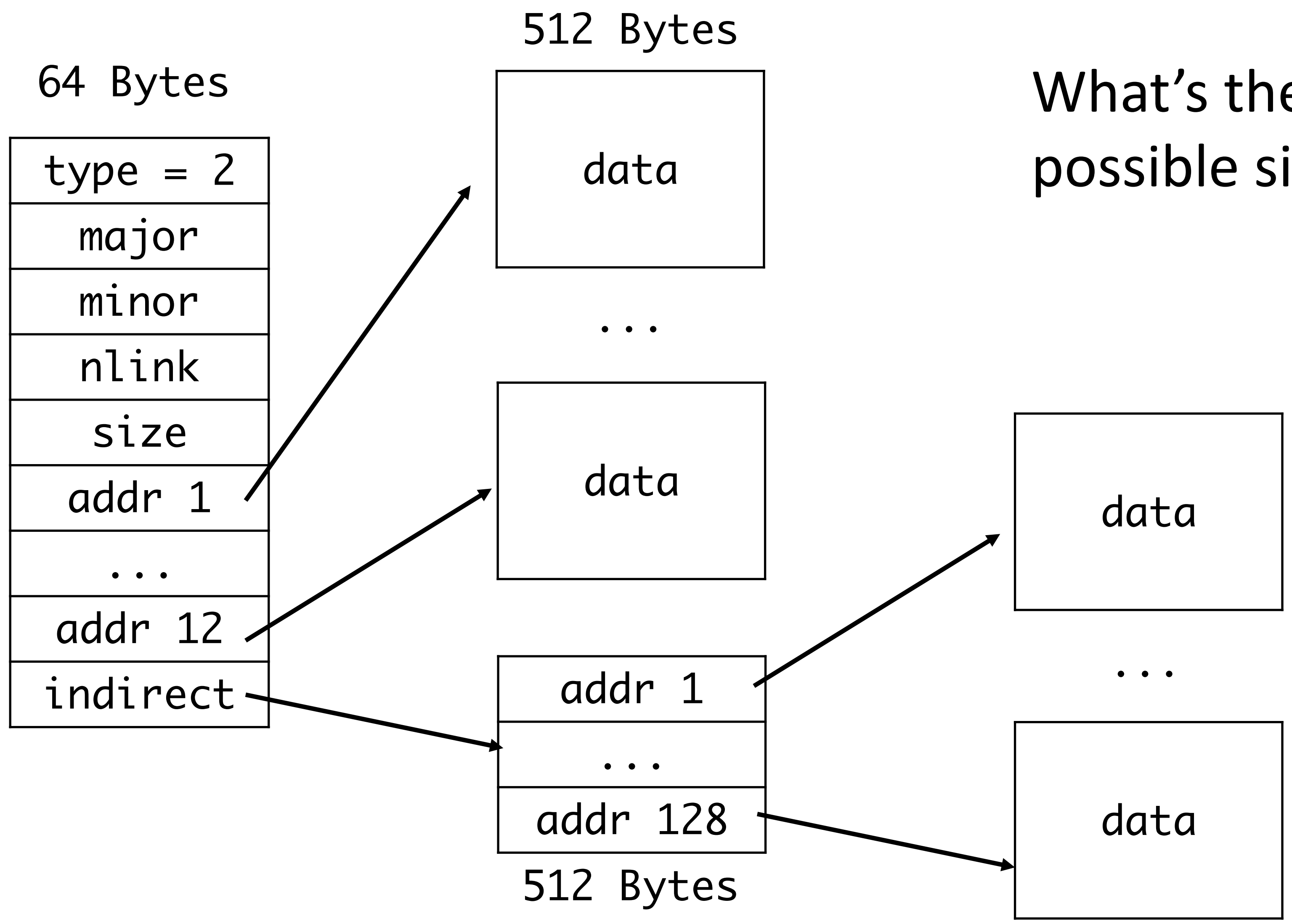


```
struct dirent {
  ushort inum;
  char name[DIRSIZ];
};
```



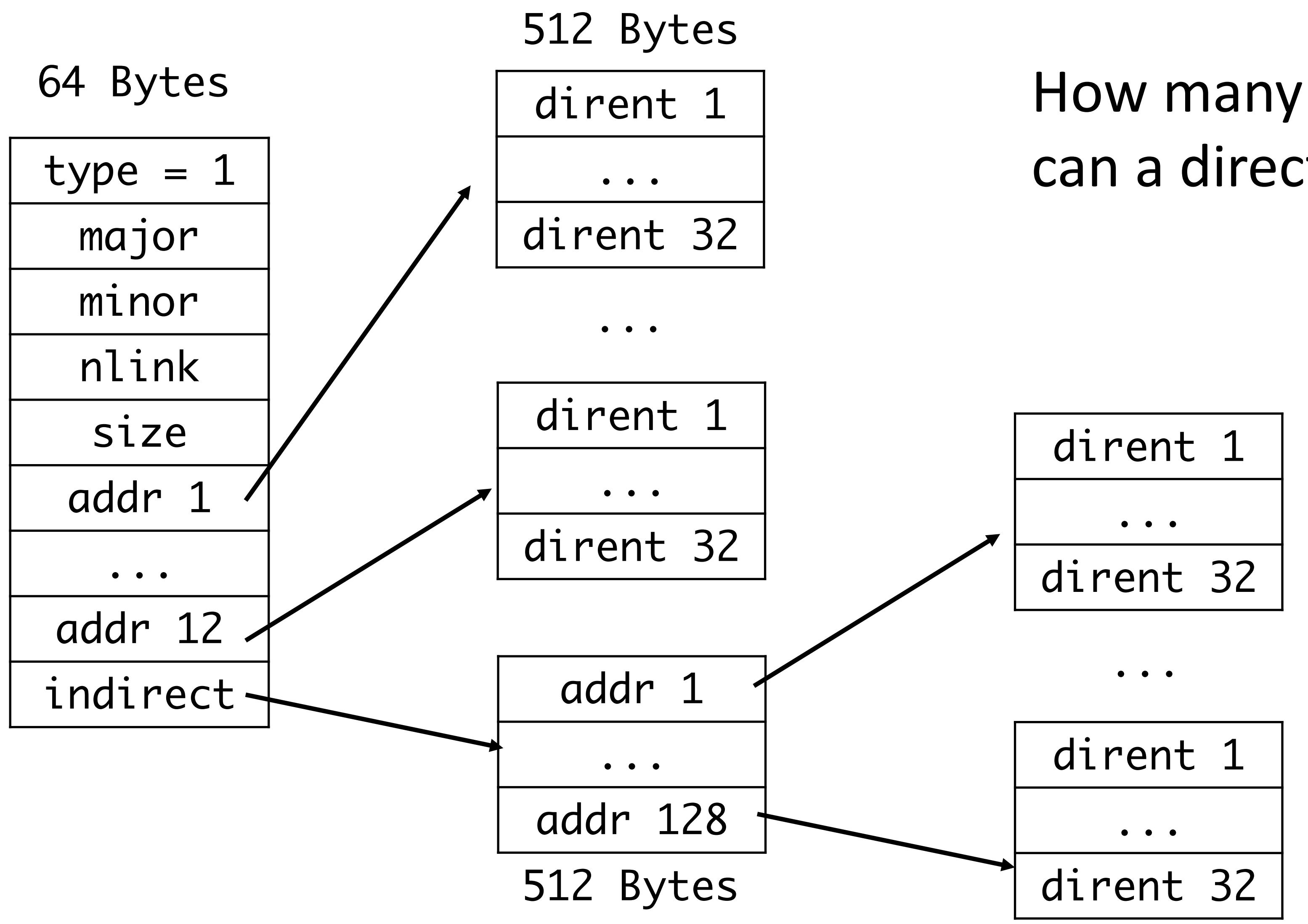
```
// On-disk inode structure
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+1];   // Data block addresses
};
```

# Inode of a regular file in xv6



What's the maximum possible size of a regular file?

# Inode of a directory in xv6



How many files/subdirectories can a directory have at most?

# Bitmap in xv6

- Each bit in the bitmap is associated with a block, NOT an inode.
- Although the very first block is unused, it is always marked as 1 in the first bit of the bitmap, and so are all the blocks where the inodes and bitmap itself reside.
- Bitmap is grouped in byte.
- Intel x86 processors use little-endian.
- Example:
  - `ff c2 => 1111 1111 1100 0010`

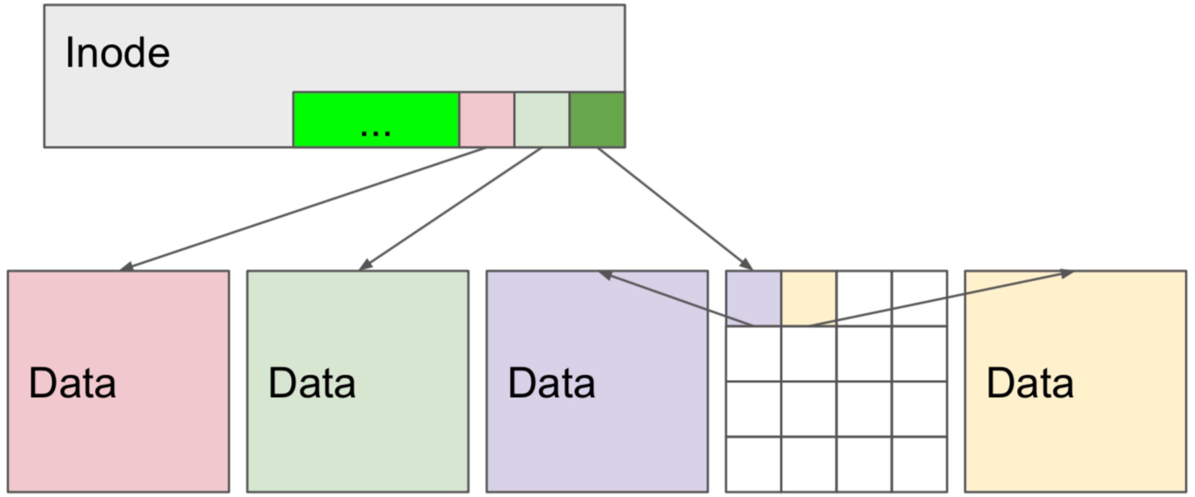
```
           7 6 5 4 3 2 1 0 | 15 14 13 12 11 10 9 8
```

## Demos

- How to build your own file system image and reflect in xv6
  - Closer look at `mkfs.c`
- How xv6 files change the image
  - Closer look at `fs.img` with `xxd`
  - Some tricks editing file images with vim:
    - `:%!xxd` to open the image; `:%!xxd –r` to save changes
- How to read image? – `mmap()`!

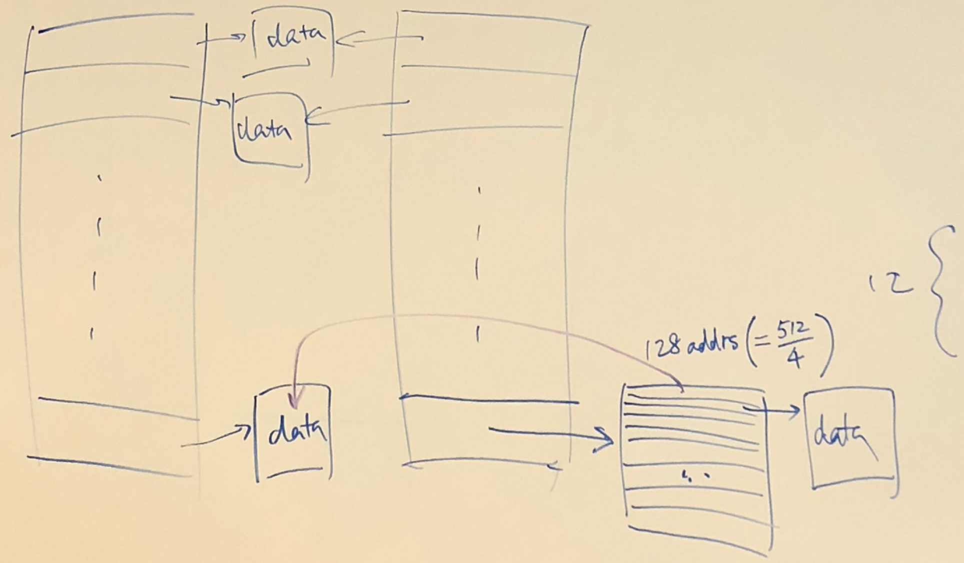
## Demos

- Using GDB, go through the entire write system call.
  - Key methods to understand:
    - `writei()` in `kernel/sysfile.c`
    - **`bmap()`** in `kernel/fs.c`
    - `bread()`, `bwrite()` in `kernel/bio.c`
- How does xv6 normally handle large files?
- What do we do to handle small files here?



≤ 13 Blocks new:

bmap( );


data
data
data
128 addrs $(=\frac{512}{4})$
data
12